**Walkthrough: Creating the Data Access and Business Logic Layers in ASP.NET**

When you work with data in ASP.NET, you will benefit by using common software patterns. One of these patterns is to separate the data-access code from the business-logic code that governs access to the data or that provides other business rules. In this pattern, these two layers are separate from the presentation layer, which consists of the pages that the Web site user accesses to view or change data.

A Visual Studio Web site project with source code is available to accompany this topic: [Download](http://go.microsoft.com/fwlink/?LinkId=192426).

ASP.NET can provide separation between data access, business logic, and presentation in several ways. For example, the data source model, which includes server controls such as the [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) and [ObjectDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.objectdatasource.aspx) controls, separates the presentation layer from the data-access code and the business logic.

Another pattern is to include the data-access logic directly in the ASP.NET pages (the presentation layer). For example, you might write ADO.NET code in the ASP.NET page's code-behind page or use the [SqlDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.aspx) control. This approach tightly couples the data-access logic with the presentation layer.

The recommended approach is to separate the data access logic from the presentation layer. This separate layer is referred to as the data-access layer. The data-access layer can be implemented as a separate Class Library project. However, you can also use tools in Visual Web Developer that can generate a data-access layer for you.

|  |
| --- |
| **NoteNote** |
| You cannot create a class library project in Microsoft Visual Web Developer Express. However, you can create a separate project by using Visual Basic Express or Visual C# Express and then include the output from that class as an assembly (DLL) in the Web site. |

If your Web site displays or updates data, you should create a data-access layer and business-logic layer before creating the user interface.

A data-driven Web application usually includes a data-access layer by using typed datasets or entity classes that represent the data. It also includes a business-logic layer that enforces custom business rules. Finally, it includes a presentation layer by using ASP.NET pages, and by using master pages and themes to create a common page layout. This walkthrough shows how to create a data-access layer.

The data-access layer includes all the code that is specific to the underlying data source. This includes code that creates a connection to the database and that issues Select, Insert, Update, and Delete commands. The data-access layer typically contains classes that implement methods for accessing the underlying database data. The presentation layer does not directly work with data. Instead, it invokes classes and methods in the data-access layer for all data requests. You can customize these classes using your own business logic.

Tasks illustrated in this walkthrough include the following:

* Creating a SQL database and adding data.
* Adding a **LINQ to SQL** file that acts as the data-access layer.
* Creating a page that works as the presentation layer.
* Adding a [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control to a page that communicates between the presentation layer and the data-access layer.

http://i.msdn.microsoft.com/Global/Images/clear.gif Prerequisites

To complete the walkthrough, you will need the following:

* Visual Studio or Visual Web Developer Express installed on your computer.

|  |
| --- |
| **NoteNote** |
| If you are using Visual Studio, the walkthrough assumes that you selected the **Web Development** collection of settings when you started Visual Studio the first time. For more information, see [How to: Select Web Development Environment Settings](http://msdn.microsoft.com/en-us/library/ff521558.aspx). |

* SQL Server Express. If you have Microsoft SQL Server installed, you can use that instead.

http://i.msdn.microsoft.com/Global/Images/clear.gif Creating A Web Site

The first step is to create a Web site.

This walkthrough uses a Web site project. You could use a Web application project instead. For information about the difference between these Web project types, see [Web Application Projects versus Web Site Projects](http://msdn.microsoft.com/en-us/library/dd547590.aspx).

**To create a new file system Web site**

1. Open Visual Studio or Visual Web Developer Express.
2. In the **File** menu, click **New Web Site**.

The **New Web Site** dialog box is displayed.

1. Under **Installed Templates**, click **Visual Basic** or **Visual** **C#** and then select **ASP.NET Web Site**.
2. In the **Web** **location** box, select **File System**, and then enter the name of the folder where you want to keep the pages for your Web site.

For example, type the folder name **C:\WebSites**.

1. Click **OK**.

Visual Studio creates a Web site project that includes prebuilt functionality for layout (a master page, the Default.aspx and About.aspx content pages, and a cascading style sheet), for Ajax (client script files), and for authentication (ASP.NET membership).

http://i.msdn.microsoft.com/Global/Images/clear.gif Connecting to a Database

The next step is to connect to a database in Visual Web Developer by using the **Server Explorer** window. (In Visual Web Developer Express, the window is named **Database Explorer**.) Creating a connection to a database in **Server Explorer** lets you add tables, stored procedures, views, and other database elements, all within Visual Studio. You can also view table data or create queries manually or by using the **Query Builder** window.

When you build the typed dataset for the data-access layer later in this walkthrough, you must create a connection in Visual Web Developer to the database. You can provide the connection information manually. However, Visual Web Developer can simplify this process because it automatically populates a list of the databases that are already registered in **Server Explorer**.

For this walkthrough you will create a new database for tracking task items.

http://i.msdn.microsoft.com/Global/Images/clear.gif Creating a New SQL Server Express Database

In this section you will create a new SQL Server Express database that will store task information for a to-do list.

**To add a database to the Web site**

1. In **Solution Explorer**, right-click the name of the Web site and then click **Add New Item**.

The **Add New Item** window is displayed.

1. Select **SQL Database** and name the database **Tasks.mdf**.
2. Click **OK**.
3. When Visual Web Developer asks you whether the database should be stored in the App\_Data folder, click **Yes**.

**Creating a Schema and Sample Data for the Tasks Database**

You can use the database design and editing features to create a schema for the table that stores the task items.

**To define a schema and add data for the Tasks database**

1. In **Solution Explorer**, open the **App\_Data** folder and double-click Tasks.mdf.

The Tasks database node is opened in **Server Explorer**.

1. Right-click the **Tables** folder and then click **Add New Table**.

The table definition window is displayed.

1. Create the following columns in the table:

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Properties** |
| taskId | int | Not null |
| name | nvarchar(50) | Not null |
| dateCreated | datetime | Not null |
| isComplete | bit | Not null |

1. Select the row for the **taskId** column, right-click the row, and then click **Set Primary Key**.
2. In the **Properties** window, set the **Identity Column** property to the **taskId** column.
3. Save the table, name it **TasksList**, and then close the table-definition window.
4. Right click the table in **Server Explorer** and then click **Show Table Data**.

An editing window is displayed where you can view, add, and edit the data.

1. Add four or five records to the table.

You do not have to specify a value for the **taskId** column, because it is an identity column and its value is automatically assigned.

1. Close the editing window.

http://i.msdn.microsoft.com/Global/Images/clear.gif Creating the Data Access and Business Logic Layers

You can create a data-access and business-logic layer for the database that you just created in several ways. You can customize these classes using your own business logic.

In this walkthrough, you will create a class that represents database entities. You can then add your own business logic to these generated classes. (You will not add business logic to the classes in this walkthrough.)

In this walkthrough, you will use Language Integrated Query (LINQ) to work with data. LINQ applies the principles of object-oriented programming to relational data. It provides a unified programming model for querying and updating data from different types of data sources and extends data capabilities directly into the C# and Visual Basic languages. For more information about LINQ, see [LINQ (Language-Integrated Query)](http://msdn.microsoft.com/en-us/library/bb397926.aspx).

You will use LINQ to SQL classes as the data-access layer. You will use the **Object Relational Designer** window in Visual Web Developer to generate entity classes that represent the data.

**Mapping the Tasks Database to a SQL Data Context Class**

To begin creating the data-access layer, you add a typed dataset to the project.

**To create a class for the Tasks table**

1. If the Web site does not already have an App\_Code folder, right-click the project in **Solution Explorer**, click **Add ASP.NET Folder**, and then click **App\_Code**.
2. Right-click the App\_Code folder and then click **Add New Item**.

The **Add New Item** dialog box is displayed.

1. Under **Visual Studio installed templates**, select the **LINQ to SQL Classes** template and rename the file **Tasks.dbml**.
2. Click **Add**.

The **Object Relational Designer** window is displayed.

1. In **Server Explorer**, drag the TasksList table into the **Object Relational Designer** window.
2. Save the Tasks.dbml file.

Visual Web Developer creates the Tasks.dbml.layout file in the **App\_Code** folder under Tasks.dbml. It also creates either Tasks.designer.cs or Tasks.designer.vb, depending on what programming language you are working with.

1. In **Solution Explorer**, open the Tasks.designer.cs or Tasks.designer.vb file.

Notice that the code contains classes named TasksDataContext and TasksList. The TasksDataContext class represents the database, and the TasksList class represents the database table. The parameterless constructor for the TasksDataContext class reads the connection string for the database from the Web site's configuration file (Web.config).

1. Open the Web.config file.

Notice that a connection string to the Tasks database has been added in the **connectionStrings** element.

1. Close the class file and the Web.config file.

http://i.msdn.microsoft.com/Global/Images/clear.gif Creating and Configuring a LinqDataSource Control

Now that you have a database table and classes that represent database entities, you can use a [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) on an ASP.NET Web page to access the database. The [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control makes LINQ available to Web developers through the ASP.NET data-source control architecture.

The [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control creates the code for selecting, inserting, updating, and deleting objects in the database. These classes can be called by business logic to perform database functions and apply business-logic rules.

**To create and configure a LinqDataSource control**

1. Open or switch to the Default.aspx page.
2. Switch to **Design** view.
3. From the **Data** tab of the **Toolbox**, drag a [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control onto the Web page.

You can leave the ID property as **LinqDataSource1**.

1. In the **LinqDataSource Tasks** smart tag panel, click **Configure Data Source**.
2. In the context object list, select **TasksDataContext** and then click **Next**.
3. In the list, select **TasksLists(Table<TasksList>)**, and then click **Finish**.
4. In the **LinqDataSource Tasks** smart tag panel, select the **Enable Delete**, **Enable Insert**, and **Enable Update** check boxes.
5. Save the page.

Notice that you did not have to specify any database commands for selecting the data.

http://i.msdn.microsoft.com/Global/Images/clear.gif Creating and Configuring a GridView Control

To create a user interface for the data that is made available by the [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control, you can use various data controls. In this walkthrough you will add a [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control to the page in order to view, update, and edit the data in the TasksList table.

**To create and configure a GridView control**

1. In the Default.aspx page, switch to **Design** view.
2. From the **Data** tab of the **Toolbox**, drag a [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control onto the Web page
3. In the **GridView Tasks** smart tag panel, select **LinqDataSource1** as the data source. (If you assigned a different name when you created the [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control, use that name.)
4. In the **GridView Tasks** smart tag panel, select the **Enable Editing** and **Enable Deleting** options.
5. Press CTRL+F5 to run the page.

The page displays the data that you entered previously in the walkthrough, and it enables you to edit or delete rows.

http://i.msdn.microsoft.com/Global/Images/clear.gif Next Steps

This walkthrough has illustrated how to create the data-access and business-logic layer of an application by using the **LINQ to SQL Classes** template and the [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) and [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) server controls. You created a way for Web site pages to access data that is flexible and that is not directly tied to the presentation layer of the Web site.

The topic [Walkthrough: Creating an Ajax-Enabled Data Application](http://msdn.microsoft.com/en-us/library/bb763182.aspx) uses the [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) control to create an AJAX-enabled Web application that displays and updates information in the Tasks database.